Problem Set #3—Functions

Due: Wednesday, October 2

Problem 1 (Chapter 4, exercise 2, page 154)

Heads...
Heads...
Heads...

A weaker man might be moved to re-examine his faith, if in nothing else at least in the law of probability.

—Tom Stoppard, Rosencrantz and Guildenstern Are Dead, 1967

Write a function `consecutiveHeads(numberNeeded)` that simulates tossing a coin repeatedly until the specified number of heads appear consecutively. At that point, your program should display a line on the console that indicates how many coin tosses were needed to complete the process. The following console log shows one possible execution of the program:

```
ConsecutiveHeads
Tails
Heads
Heads
Heads
Tails
Heads
Heads
Heads
It took 7 tosses to get 3 consecutive heads.
```

Problem 2

These two exercises come from last year’s practice or actual exams. You can get the right answer by running them in Python, but the point is to work these out by hand as you would on the midterm or final.

2a) What is the result of calling the function `slytherin(9)` given the following code:

```python
def slytherin(k):
    return ravenclaw(k, 2)

def ravenclaw(n, k):
    if k == 0:
        return 0
    else:
        return ravenclaw(n + 1, k - 1) + hufflepuff(n, 3)

def hufflepuff(x, n):
    if n == 0:
        return 1
    else:
        return x * hufflepuff(x, n - 1)
```
2b) What is the result of calling the function `enigma()` given the following code:

```python
def enigma():
    def puzzle(x):
        def riddle(y):
            return 2 * x - y
        return riddle
    x = 10
    y = 37
    f = puzzle(y)
    return f(x)
```

**Problem 3**

Write a recursive function `raiseToPower(x, k)` that computes $x^k$ using the following recursive definition:

$$x^k = \begin{cases} 
1 & \text{if } k = 0 \\
 x \times x^{k-1} & \text{otherwise}
\end{cases}$$

The parameter $x$ may be any number, but you may assume that the parameter $k$ is a nonnegative integer.

**Problem 4**

Rewrite `raiseToPower(x, k)` from Problem 3 so that it uses the following recursive strategy:

$$x^k = \begin{cases} 
1 & \text{if } k = 0 \\
 (x^{k/2})^2 & \text{if } k \text{ is even} \\
 (x^{k/2})^2 \times x & \text{if } k \text{ is odd}
\end{cases}$$

This strategy is substantially more efficient. For example, computing $x^{1024}$ by the method in problem 3 would take 1024 multiplications but only 10 using this approach.

**Problem 5 (Chapter 4, exercise 13, page 158)**

Rewrite the `gcd` function that uses Euclid’s algorithm shown on page 67 so that it computes the greatest common divisor recursively using the following rules:

- If $y$ is zero, then $x$ is the greatest common divisor.
- Otherwise, the greatest common divisor of $x$ and $y$ is always equal to the greatest common divisor of $y$ and the remainder of $x$ divided by $y$. 